System Test Report

STORC DASHBOARD PROJECT

CD Jam

CSC 191 – FALL 2015 | 11/XX/2015

Table of Contents

1. INTRODUCTION

This is the Software Test Report document for the STORC Dashboard Project sponsored by Dr. Michael Christensen.

This project is being undertaken by the CD Jam software development team. The team is comprised of undergraduate students majoring in Computer Science at California State University, Sacramento. The team members are enrolled in a two-semester senior project course required of all undergraduate majors. Successful delivery of the desired software product will fulfill the senior project requirement for the student team members.

PROJECT SPONSOR:
Name: Michael Christensen
Title: Assistant Vice President for Risk Management Services & Director of STORC
Organization Name: Sustainability Technology Optimization Research Center (STORC)
Contact Information:
Phone: (916) 278-5252
Email: storc@csus.edu

CD JAM DEVELOPMENT TEAM:
Name: Cole Culler
Contact Information:
Phone: (530) 575-7683
Email: cbroski.culler@gmail.com
Name: David Grapentine
Contact Information:
Phone: (707) 471-8749
Email: davidjoaograpentine@gmail.com
Name: Ashley Gregory
Contact Information:
Phone: (209) 304-1884
Email: aa5gregory@gmail.com
Name: John Jones
Contact Information:
Phone: (916) 475-8460
Email: felixequal@gmail.com
Name: Michael Smith
Contact Information:
Phone: (916) 842-8339
Email: infinitlyill@gmail.com

1.1 Purpose

The System Test Report summarizes the results of the designated testing activities identified in the System Test Specification document and provides evaluations based on these tests.

1.2 Scope

The System Test Report includes only the results and evaluations from testing the implementation of each of the system's Use Cases. The testing reported is that which was specified in the System Test Specification document.

1.3 Definitions, Acronyms, and Abbreviations

1.3.1 Definitions

Administrator: A super user that oversees other users and the projects within STORC.
Aquaponics: A cycle between hydroponically grown plants and aquatic animals, in which the waste produced from animals supplies nutrients for plants which in turn purifies the water.
Biodiesel: A substitute for diesel created by a biological chemical reaction.
Dashboard: A collection of data laid out in an easy to read format represented in a graphical format.
Pass/fail Criteria: Decision rules used to determine whether a software item or a software feature passes or fails a test.
Photovoltaic Cell: A device that delivers an electric current as a result of a chemical reaction from the rays of the sun.
Principal Investigator: A user that in charge one or more STORC projects and oversees one or more STORC Technicians. Principal Investigators are most often faculty or staff members at CSUS.
Public: Any user that is not directly involved with STORC or STORC activities.
Software Problem or Test Incident Report: A document reporting on any event that occurs during the testing process which requires investigation.
Technician: A user who works on a project overseen by a PI. This user generally monitors and collects data directly from one or more project stations. These are usually student volunteers.
Test Case Specification: A document specifying inputs, predicted results, and a set of execution conditions for a test item.
Test Design Specification: A document specifying the details of the test approach for a software feature or combination of software features (i.e. a Use Case) and identifying the associated tests.
Test Item: A software component (or components) that are an object of testing.
Test Log: A chronological record of relevant details about the execution of tests.
Test Plan: A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features (i.e. Use Cases) to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.
Test Procedure Specification: A document specifying a sequence of actions for the execution of a test.

Test report: A document summarizing testing activities and results. It also contains an evaluation of the corresponding test items.
Vermiculture: The cultivation of worms used for composting materials.
Webmaster: This user is in charge of maintaining and updating the default web page and widgets for the STORC Dashboard Project.

1.3.2 Acronyms

CSc – Computer Science
CSUS – California State University, Sacramento
ECS – College of Engineering and Computer Science
ERD – Entity Relationship Diagram
GUI – Graphical User Interface
HTML – Hyper Text Markup Language
IRT – Information Resources and Technology
IT – Information Technology
MySQL – My Structured Query Language
PI – Principal Investigator
PMP – Software Project Management Plan
SDS – Software Design Specification
SRS – Software Requirements
STR – System Test Report
STORC – Sustainability Technology Optimization Research Center
STS - Software Test Specification
UM – User Manual
UML – Unified Modeling Language
WCM – Web Content Management

1.3.3 Abbreviations

Admin – Administrator
CSc 190: Computer Science Senior Project - Part 1
CSc 191: Computer Science Senior Project - Part 2
Tech – Technician

1.4 References

Buckley, Bob. *CSc 190-01 Senior Project: Part 1*. CSUS, Dec. 2014. Web. 22 February 2015. http://athena.ecs.csus.edu/~buckley/CSc190/CSc190.html

STORC. CSUS STORC. n.p. Web. 22 February 2015. http://www.csus.edu/storc/about.html

1.5 Overview of Contents of Document

Section 2. SUMMARY contains the summary evaluation of the test results for each of the implemented Use Cases. *Section 3. VARIANCES* describes whatever variances might exist between the items tested and their design specification and/or requirements specifications. *Section 4. COMPREHENSIVENESS OF THE TESTING PROCESS* provides an evaluation of the comprehensiveness of the testing process against the testing criteria specified in the system test plan contained in the System Test Specification document. *Section 5. APPROVALS* contains the approval page. The signatures indicate approval of the system testing

process and the results as documented in this report by all team members and the project's faculty adviser. *APPENDIX A – TEST LOGS* contains the test logs for each of the testing sessions. *APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS* contains all Software Problem or Test Incident reports organized by Use Case that is, for each Use Case the reports are included in chronological order.

2. SUMMARY

2.1 Use Case 1: Manage Data

2.1.1 Associated Test Items

The test items that are associated with Use Case 1: Manage Data are the following:

- Being able to load the Edit Data tab
- Loading one of the project tab
- Having all data associated with that database load into the table
- Edit a data entry
- Canceling out of an edit data entry
- Saving the edits made to an entry to the database
- Delete a data entry
- Will not allow you to input random characters/leave blank in the input boxes

2.1.2 STS Cross Reference

These are the following sections in which the Use Case 1: Manage Data is mentioned in the STS

- 4.3: Feature 4: Review and Submit Data Interface
- 4.9 Feature 6: View and Select Pending Data Interface

2.1.3 Testing Environment

All testing was done using Microsoft Visual Studio with the web application being launched in a web browser. The Microsoft SQL database is hosted on a team member's Azure account so all members have the capability to access the database.

2.1.4 Test Log Reference

APPENDIX A – TEST LOGS in Test Log: Use Case 1 – Manage Data

2.1.5 Test Incident Reports Reference

No test Incident report done for this use case.

2.1.6 Evaluation of Testing

Once data was successfully entered into a database, the testing team would view the 'Edit Data' page for a particular project. To test the editing pages the testing team clicked the Update, Edit, and Delete buttons respectively to ensure their functionality. To ensure the Edit functionality worked correctly a row from each cell would selected and thoroughly vetted to ensure proper functionality. Delete was tested by clicking the delete button and comparing the Edit view to the values database. To test the update functionality, data was changed in the Edit view and the Update button was clicked. Then a group member would compare the database to the

Edit view. Overall the testing data sufficed. All test performed gave positive results indicating that our system was functioning correctly.

2.2 Use Case 3: Input Data

2.2.1 Associated Test Items

The test items that are associated with Use Case 3: Input Data are the following:

- Being able to load the Input Data tab
- Loading one of the project tab under Input Data
- Having validation checking
    - Boxes required not to be blank for required data
    - Data not being a different type then being asked for
    - Negative numbers not allowed
- Being able to reset form
- Submitting data from the form to the database

2.2.2 STS Cross Reference

These are the following sections in which the Use Case 3: Input Data is mentioned in the STS

- 4.3: Feature 3: Data Input Interface
- 4.6 Feature 3: Data Input Interface

2.2.3 Testing Environment

All testing was done using Microsoft Visual Studio with the web application being launched in a web browser. The Microsoft SQL database is hosted on a team member's Azure account so all members have the capability to access the database.

2.2.4 Test Log Reference

APPENDIX A – TEST LOGS in Test Log: Use Case 3 – Input Data

2.2.5 Test Incident Reports Reference

APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS in Test Incident Report – Use Case 3: Input Data

2.2.6 Evaluation of Testing

Inserting data into a webform is an enormous concern when testing input validity. There are an infinite number of possible configurations that a user could input data. The tester has to pick a dataset that would accurately reflect many of the possible strings of data. With the help of C#'s build in error checking, and a good dataset the overall result was positive. The data being entered by STORC employees are one of 3 datatypes: int; floats; strings. In order to test our webform, the tester would insert random characters into the webform and would verify that the data was entered into the database. The tester would also enter in data that is not valid on the webform and would check if the error were handled properly. This section was the most difficult in term of testing. Overall the testing done should be sufficient for STORC at this time.

2.3 Use Case 4: Customize/View Dashboard

2.2.1 Associated Test Items

The test items that are associated with Use Case 4: Customize/View Dashboard are the following:

- Being able to load the Home page tab
- Add a new widget
- Delete a new widget
- Edit a new widget
- Move widgets around
- Display the data and graph chosen by user on widget
- Graphs reading from database

2.2.2 STS Cross Reference

These are the following sections in which the Use Case 4: Customize Dashboard is mentioned in the STS

- 4.3: Feature 1: Customizing Widget
- 4.4 Feature 1: Homepage Management Interface

2.2.3 Testing Environment

All testing was done using Microsoft Visual Studio with the web application being launched in a web browser. The Microsoft SQL database is hosted on a team member's Azure account so all members have the capability to access the database.

2.2.4 Test Log Reference

APPENDIX A – TEST LOGS in Test Log: Use Case 4 – Customize/View Dashboard

2.2.5 Test Incident Reports Reference

APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS in Test Incident Report – Use Case 4: Customize/View Dashboard

2.2.6 Evaluation of Testing

To test out manage widget the creator of the widgets portion of the project tested out each single element of the widget from javascript to the appearance on the screen of the widget. To testing Adding a widget, a new widget was created by right-clicking on the side next to a widget and clicking add. Then a chart would be built on the newly created widget in order to ensure it's functionality. To test widget deletion, an existing widget would be deleted by clicking on the red X at the top of the screen or by right clicking anywhere on the widget and selecting delete. To test edit, the tester would right-click on an existing graph and change the values in the drop down boxes to complete the process. Overall went well since the only interaction the user had with our web application was with a drop down menu or a date picker. No major errors appeared in testing and our testing set sufficed for the type of inputs we have the user enter.

3. VARIANCES

There were variance between CD Jam's Test Plan in the STS compared to what testing was completed STR. Most of these variances occurred because some use cases and features CD Jam planned to implement were not completed due to time constraints, learning curves, and the lack of knowledge about the data being collected for each project hosted at STORC. In order to satisfy the requirements for the sponsor, CD Jam decided to make the web app more of a management tool for the Admin and PI users of projects thus user login, managing users, ability for public to view, saving widgets location, graph held in each widget, and widgets to be more than graphs could not be completed for this project. CD Jam only coded and tested Use Case 1 – Manage Data which included being able to edit and delete data to the database, Use Case 3 – Input Data which allowed users to input data into database, and Use Case 4 – Customize Dashboard which the user can add, edit, delete, and move widgets. Other than the previous stated changes, the test plan for the project stayed the same as the Test Plan stated in the STS.

4. COMPREHENSIVENESS OF THE TESTING PROCESS

For the parts that CD Jam was able to code, Use Case 1, 3, and 4, the testing team was only able to perform unit and interface testing. The other testing that was listed within the STS was not completed due to time constraints on this project. As for the use case and features not removed from the scope of the project, all were tested as stated in the STS for unit and interface testing.

5. APPROVALS

By signing you agree that all conditions and commitments to the project are accurate to the best of your knowledge. I certify that the information in this System Test Report is correct and the senior project group *CD Jam* can continue on with the design of the project. I also certify that I will follow and provide all needing requirements stated in this document and that I am willing to follow through with all conditions.

CD Jam Team members:

X _____
Cole Culler

X _____
David Grapentine
Project Lead

X _____
Ashley Gregory

X _____
John Jones

X _____
Michael Smith

Faculty Advisor:

X _____
Ying Jin
Faculty Advisor

APPENDIX A – TEST LOGS

Test Log: Use Case 1 – Manage Data

Use Case

Use Case 1: Manage Data

Test Items Required Testing

Review and Edit Data Page

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Starting at the homepage of the dashboard, select the '***Edit Data***' tab.
- On the Manage Data page, select each of the project tabs one at a time and look to see if the detailed view of the data appears.
- Try editing the data in the '*Fish Data'* section of the detailed view list.
- After changing some of the data within the detailed view list, select the '*Update'* button.
- Go into Visual Studio and view the data table to see if the new data has been submitted into the database.

Ending Activities

- Repeat the steps as described in the beginning activities section, but this time go through Biodiesel, Compost and Vermiculture projects.

Observable Results

- The Manage Data interface was completely functional for all projects that are currently setup for the STORC Dashboard.
- When selecting the '*Edit'* button next to the field that needs to be changed, all data within the detailed view list becomes selectable.

Test Results

This test was successful.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Test Log: Use Case 3 – Input Data

Compost Input Form

Use Case

Use Case 3: Input Data

Test Items Required Testing

Compost Data by Vendor Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Begin the testing process by selecting 'The Buzz/Ecogrounds' from the 'Waste Received From:' section.
- Select a date for the month of December from the 'Date:' section of the webform.
- Enter a single positive integer for the 'Weight' section (i.e. enter the number 3).
- Enter positive percentages for the 'Percentage Grains, Percentage Fruit, Percentage Vegetables, Percentage Dairy, Percentage Paper, and Percentage Coffee Grounds' making sure that they add up to 100%.
- Select 'Yes' for 'Trash Present'.
- Type in a single sentence into the 'Notes:' section of the webform.
- Click the Submit button.
- View the Compost Table in our database to see if the new results have been added.
- Go through steps 1 and 2 again.
- For the 'Weight:' section, enter a double digit number.
- For the percentages, enter numbers that do not add up to 100%
- Select an input for 'Trash Present:' section.
- Click the Submit button.
- View Results.

Ending Activities

- Repeat all steps as described above, but for each trial use a different input for the 'Waste Received From:' section of the webform until all inputs have been tested.
- Fill out the webform and then click the reset button. View Results.
- Next, try filling out the webform with invalid data, such as negative values and non-integers.

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- All data that was correctly entered was submitted to the database successfully.

- If the percentages entered do not add up to 100%, an error message appears notifying the user of the error. This is expected behavior for the webform.
- If a negative integer is entered, an error message appears notifying the user to try again. This is expected behavior.

Failed Results

- There is no message indicating that the data was submitted successfully. Also the webform does not reset once the user clicks the submit button.
- The Reset button does not clear the webform as it should.
- When entering inappropriate data such as characters where integers should be, the webform breaks because of an unhandled exception. This occurs in all webforms.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Hot Compost Webforms

Use Case

Use Case 3: Input Data

Test Items Required Testing

Hot Compost Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Click on the Hot Compost tab of the Data Input section of the Dashboard.
- Select a date for the month of December.
- Enter a positive integer for the 'Weight:' section of the webform.
- Click the 'Submit' button.
- Refresh the Compost Data Table inside of Visual Studio and see if the new data was entered successfully.

Ending Activities

- For each trial, enter inappropriate data into the 'Weight:' section of the webform. Some examples include:
  - Negative integers
  - Decimals
  - Characters instead of integers
  - 9+ digits

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- All data submitted was successfully added to the Compost database.
- Any negative integers were detected and an error message notifies the user.
- All integers with 9+ figures is submitted successfully into the Compost database.

Failed Results

- When entering characters into the 'Weight' section, an unhandled exception error occurs.
- After selecting the date, if the user clicks back on the date and accidentally types something else in, a system format exception error occurs.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Vegetable Oil Compost Webform

Use Case

Use Case 3: Input Data

Test Items Required Testing

Vegetable Oil Compost Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Add valid data into the vegetable oil webform.
- Click the 'Submit' button and observe results.
- Look at the database to see if the data was submitted successfully.

Ending Activities

- For each trial, enter inappropriate data into the 'Weight' and 'Percentage Full:' section of the webform. Some examples include:
    - Negative integers
    - Decimals

    - Characters instead of integers
    - 9+ digits

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- Data was successfully submitted into the database.
- Any negative integers or invalid entries are caught by the system and the user is notified of the error.

Failed Results

- The 'Reset' button does not work.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Biodiesel Webform

Use Case

Use Case 3: Input Data

Test Items Required Testing

Biodiesel Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Enter valid data for Biodiesel webform.
- Click the 'Submit' button.
- View results.
- Check to see if the data submitted is actually in the Biodiesel database.
- Repeat the steps 1 through 4 two more times.

Ending Activities

- Enter invalid data into the Biodiesel webform. Invalid data includes the following:
    - Negative integers
    - Decimals
    - Characters instead of integers
    - 9+ digits
- Tried entering the same batch number as a number already entered in the database.

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- Valid data is submitted successfully into the Biodiesel database.
- Negative values are caught by the program and the user is notified.
- Empty values are caught and the user is notified.
- Decimals are accepted.
- When entering the same batch number as one that is already submitted into the database, the database re-numbers the batch number to the appropriate item. (It may be nice to notify the user with a warning or a message).

Failed Results

- The 'Reset' button does not appear to have any functionality. It does not clear the text-fields as it should.
- There is no message letting the user know that the data was submitted successfully.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Plant Production Webform

Use Case

Use Case 3: Input Data

Test Items Required Testing

Plant Production Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Enter valid data for the Aquaponics webform.
- Click the 'Submit' button.
- View results.
- Check to see if the data submitted is actually in the AquaFarm database.
- Repeat the steps 1 through 4 two more times.

Ending Activities

- Enter invalid data into the Aquaponics webform. Invalid data includes the following:
  - Negative integers
  - Decimals
  - Characters instead of integers
  - 9+ digits

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- Valid data is submitted successfully into the AquaFarming database.
- Negative values are caught by the program and the user is notified.
- Empty values are caught and the user is notified.
- Decimals are accepted.

Failed Results

- The 'Reset' button does not appear to have any functionality. It does not clear the text-fields as it should.
- There is no message letting the user know that the data was submitted successfully.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Fish Production Webform

Use Case

Use Case 3: Input Data

Test Items Required Testing

Fish Production Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Enter valid data for the Fish Production webform.
- Click the 'Submit' button.
- View results.

- Check to see if the data submitted is actually in the AquaFish database.
- Repeat the steps 1 through 4 two more times.

Ending Activities

- Enter invalid data into the Aquaponics webform. Invalid data includes the following:
  - Negative integers
  - Decimals
  - Characters instead of integers
  - 9+ digits

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- Valid data is submitted successfully into the AquaFish database.
- Negative values are caught by the program and the user is notified.
- Empty values are caught and the user is notified.
- Decimals are accepted.

Failed Results

- The 'Reset' button does not appear to have any functionality. It does not clear the text-fields as it should.
- There is no message letting the user know that the data was submitted successfully.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Aquaponics Webform

Use Case

Use Case 3: Input Data

Test Items Required Testing

Aquaponics Webform

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Enter valid data for the Aquaponics webform.

- Click the ‘Submit’ button.
- View results.
- Check to see if the data submitted is actually in the AquaFish database.
- Repeat the steps 1 through 4 two more times

Ending Activities

- Enter invalid data into the Aquaponics webform. Invalid data includes the following:
  - Negative integers
  - Decimals
  - Characters instead of integers
  - 9+ digits

Observable Results

- Some parts of the program ran as expected and the program did not crash, while other pieces of the program did not function as expected.

Successful Results

- Valid data is submitted successfully into the AquaTank database.
- Negative values are caught by the program and the user is notified.
- Empty values are caught and the user is notified.
- Decimals are accepted.

Failed Results

- The ‘Reset’ button does not appear to have any functionality. It does not clear the text fields as it should.
- There is no message letting the user know that the data was submitted successfully.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Test Log: Use Case 4 – Customize/View Dashboard

Creating a Widget

Use Case

Use Case 4: Customize/View Dashboard

Test Items Required Testing

Creating Widget

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the

project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Click on the drop down menu next to '*Select the type Chart for this Widget*'. This dropdown menu is located at the top of the widget.
- Select a project in the next drop down box that appears.
- Select the data that we want displayed for that widget.
- Click the '*Create Chart'* button.
- Observe results.
- Repeat steps for all widget types using the same data as before. Test all data with the following chart types:
  - Bar Chart
  - Donut Chart
  - Line Chart
  - Pie Chart
- Repeat steps again for all available projects and data.
- Compare data in widget to the data in the data table.

Ending Activities

- Right click on a widget that has already been created.
- Select '*Edit Widget*'.
- Make a random change to the widget, (i.e. change project type, data type, widget style).
- Compare the data in the widget to the data in the tables.
- Repeat steps for five more widgets.

Observable Results

- Each widget was created without any problems.
- All data in the widgets matched the data in the databases.

Test Results

- There needs to be a title for each widget created.

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Deleting a Widget

Use Case

Use Case 4: Customize/View Dashboard

Test Items Required Testing

Deleting Widget

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Open a new dashboard.
- Click on the red x located at the upper right hand corner of a non-customized widget.
- Observe results.
- Create 2 more widgets, this time customize them to specific projects.
- Click on the red x located at the upper right hand corner of these widgets.
- Observe results.

Ending Activities

- Create five more widgets.
- Right click on the widgets.
- Select '*Delete Widget'* from the menu that appears.
- Observe results.

Observable Results

- For every trial, the widgets deleted without any problems.

Test Results

Successful

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Adding a Widget

Use Case

Use Case 4: Customize/View Dashboard

Test Items Required Testing

Adding Widget

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Right click anywhere in the dashboard beneath the tabs.
- Select '*Add Widget'*.
- Observe results.

- Try modifying the newly added widget.
- Observe results.
- Repeat process five more times, each time right clicking on a different location of the dashboard.

Ending Activities

- Try deleting all of the widgets on the dashboard.
- Add two widgets and assign them to specific data.
- Observe the results and compare the data on the widget to the actual data in the data tables

Observable Results

- One important thing to note is that the arrow must be towards the top of the screen in order to right click and add a widget. For example, if there is only one widget on the dashboard and the user right clicks at the bottom of the dashboard, nothing will happen. The user must right click next to the widget that already exists. If there is no widget on the dashboard, just be sure to right click near the top, below the tabs.

Test Results

Successful

Unexpected Results

None

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Moving a Widget

Use Case

Use Case 4: Customize/View Dashboard

Test Items Required Testing

Moving Widget

Team Members Doing Testing

Cole Culler and Michael Smith

Description of Testing Environment

All testing was done using Microsoft Visual Studio and/or a web browser. There are no hardware requirements needed since the project is performed solely on the tester's web-browser or within our coding IDE, Microsoft Visual Studio.

Beginning Activities

- Move the arrow over a new widget.
- Wait for the arrow to change into four arrows pointing in opposite directions.
- Once the arrow changes, hold down the left mouse button and move the widget around.

Ending Activities

- Repeat process for five more widgets

- Try moving the widget with the cursor in a different part of the widget.

Observable Results

- If you have the cursor at the bottom of the widget, the cursor changes into the four arrows pointing in opposite directions, however you are not able to move the widget. If the user tries to move the widget from the bottom of the widget, it will merely highlight the dashboard. The user is only able to move the widget if the arrow is near the top of the widget, next to the red x.

Test Results

- The cursor move icon should only be visible when the cursor is at the top of the widget.

Unexpected Results

Sometimes I am unable to move the widget if the cursor is not in the correct location.

Software Problem(s) or Test Incident Reports(s)

Please see APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

APPENDIX B – SOFTWARE PROBLEM or TEST INCIDENT REPORTS

Test Incident Report – Use Case 3: Input Data

Inputs

- Select Input Data Tab
- Select a Project
- Fill out the webform for that particular project, all with valid entries.
- Select one of the input fields, and purposefully enter a character instead of an integer.

Expected Results

- Each webform should have proper Exception handling in order to catch illegal characters.
- The user should be notified about his or her mistake with an error message of some kind.

Actual Results

- Format Exception was unhandled by user code.
- Error: UnhandledException
- The entire Dashboard stops working.

Anomalies

N/A

Date and Time

12/15/2015 3:53 PM

Procedures Followed

- Test Incident report created
- Coder will be notified of problem and incident report

Environment

- Windows 7
- Visual Studio Enterprise 2015
- Web Browser used was Google Chrome

Attempts to rerun test

We ran the test five times in a row, all with different webforms. Every trial resulted in a Format Exception error.

Tester and Author of the Report

Cole Culler and Michael Smith

Analysis of the Incident/Problem

- Input validation still needs further development for all webforms in order to resolve the issue.
- It is important to note that the inputs and results mentioned above are the same for all Input Webforms.

Suggested Remedy/Fix

- Apply catch statements for each webform in order to catch any illegal characters that may be entered into the webforms.
- The same arguments must be applied to all webforms.

Test Incident Report – Use Case 4: Customize/View Dashboard

Inputs

- Select '*Donut Chart*' or '*Pie Chart'* from the first dropdown when creating a new widget.
- Select '*Aquaponics'* from the dropdown list of available projects.
- Select '*Chemical Levels*' from the dropdown list of available data points.
- Click on the '*Create Chart'* button

Expected Results

After clicking on the '*Create Chart'* button, a donut chart should appear displaying the data for Chemical Levels.

Actual Results

The following error occurs and can be read from the console: 'NullReferenceException was unhandled by user code'

Anomalies

This doesn't appear to be an anomaly. The testing team is able to recreate the same results every time.

Date and Time

12/09/2015 at 1:00 PM

Procedures Followed

- Test Incident report created
- Coder notified of problem and Test Incident report

Environment

- Windows 7
- Visual Studio Enterprise 2015
- Web Browser used was Google Chrome

Attempts to rerun test

None

Tester and Author of the Report

Cole Culler and Michael Smith

Analysis of the Incident/Problem

- Because of limited time, some of the database queries have not been completed for particular STORC projects. Because some of this functionality has not been developed, some of the charts for the widgets cannot be created.

Suggested Remedy/Fix

Unclear how to fix this issue, other than completing the queries and functionality for all STORC projects and databases.